BEYER WEAVER & THOMAS, LLP
INTELLECTUAL PROPERTY LAW
590 W. El Camino Real, Mountain View, CA 94040
Telephone: (650) 961-8300 Facsimile: (650) 961-8301
www.beyerlaw.com

FACSIMILE COVER SHEET

December 16, 2005

Receiver: **Examiner Erci B Kiss - Group Art Unit 2122**

TEL #: (571) 272-3699

FAX # : **(571) 273-8300**

Sender: Alan S. Hodes

Our Ref. No.: SUN1P253

Re: 09/467,387

Pages Including Cover Sheet(s): 23

MESSAGE:

The attached Appeal Brief and Transmittal was filed by FAX on 04/21/05, but has not been entered into the system (PAIR).

Auto-Reply Facsimile Transmission

TO: Fax Sender at 16509618301

Fax Information
Date Received: 4/21/2005 5:37:39 PM [Eastern Daylight Time]
Total Pages: 21 (including cover page)

ADVISORY: This is an automatically generated return receipt confirmation of the facsimile transmission received by the Office. Please check to make sure that the number of pages listed as received in Total Pages above matches what was intended to be sent. Applicants are advised to retain this receipt in the unlikely event that proof of this facsimile transmission is necessary. Applicants are also advised to use the certificate of facsimile transmission procedures set forth in 37 CFR 1.8(a) and (b), 37 CFR 1.6(f). Trademark Applicants, also see the Trademark Manual of Examining Procedure (TMEP) section 306 et seq.

Received Cover Page ========>

APR. 21. 2005 2:40PM 65096'830 NO. 131 P.

BEYER WEAVER & THOMAS, LLP

INTELLECTUAL PROPERTY LAW

950 W. El Camino Real, Mountain View, CA 94040
Telephone (650) 961-8300 Facsimile (650) 961-8301
www.beyerlaw.com

FACSIMILE COVER SHEET

April 21, 2005

Receiver: Examiner Eric B. Kiss
Art Unit 2121

TEL #:

FAX #: 703-872-9306

Sender: Alan S. Hodes, Reg. No. 38,185

Re: Appeal Brief Transmittal (2 pgs.)
Appeal Brief (17 pgs.)
Application No. 09/764,387
Attorney Docket No. SUNIP253

Pages Including Cover Sheet(s): 20

MESSAGE:

BEYER WEAVER & THOMAS LLP Docketed APR 25 2005 OAKLAND

DOCKETING

CONFIDENTIALITY NOTE

The information contained in this facsimile (FAX) message is legally privileged and confidential information intended only for the use of the recipient or firm named above. If the reader of this message is not the intended receiver, you are hereby notified that any dissemination, distribution or copy of this FAX is strictly prohibited. If you have received this FAX in error, please immediately notify the sender at the telephone number provided above and return the original message to the sender at the address above via the United States Postal Service. Thank you.

PAGE 02/21 * RCVD AT 4/21/2005 5:37:39 PM [Eastern Daylight Time] * SVR:USPTO-EFXRF-1/7 * DNIS:2738300 * CSID:16509618301 * DURATION (mm-ss):06-36

* * * COMMUNICATION RESULT REPORT (APR. 21. 2005 2:46PM) * * *

TTI 16509618301

TRANSMITTED/STORED: APR. 21. 2005 2:39PM

FILE MODE	OPTION	ADDRESS (GROUP)	RESULT	PAGE
131 MEMORY TX		##424756#17038729306#	OK	21/21

REASON FOR ERROR
E-1) HANG UP OR LINE FAIL
E-2) BUSY
E-3) NO ANSWER
E-4) NO FACSIMILE CONNECTION

BEYER WEAVER & THOMAS, LLP
INTELLECTUAL PROPERTY LAW
590 W. El Camino Real, Mountain View, CA 94040
Telephone: (650) 961-8300 Facsimile: (650) 961-8301
www.beyerlaw.com

FACSIMILE COVER SHEET

April 21, 2005

Receiver: **Examiner Eric B. Kiss**
Art Unit 2122

TEL #:

FAX #: **703-872-9306**

Sender: Alan S. Hodes, Reg. No. 38,185

RECEIVED
CENTRAL FAX CENTER

DEC 16 2005

BEYER WEAVER & THOMAS, LLP

INTELLECTUAL PROPERTY LAW

590 W. El Camino Real, Mountain View, CA 94040
Telephone: (650) 961-8300 Facsimile: (650) 961-8301
www.beyerlaw.com

FACSIMILE COVER SHEET

April 21, 2005

Receiver: **Examiner Eric B. Kiss**
Art Unit 2122

TEL #:

FAX # : **703-872-9306**

Sender: Alan S. Hodes, Reg. No. 38,185

Re: Appeal Brief Transmittal (2 pgs.)
Appeal Brief (17 pgs.)
Application No. 09/764,387
Attorney Docket No. SUN1P253

Pages Including Cover Sheet(s): 20

MESSAGE:

RECEIVED
CENTRAL FAX CENTER
DEC 16 2005

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Pelegri-Llopart et al.	Attorney Docket No.: SUN1P253/P4194
Application No.: 09/467,387	Examiner: Kiss, Eric B.
Filed: December 21, 1999	Group: 2122

Title: MULTI-LINGUAL TAG EXTENSION MECHANISM

CERTIFICATE OF FACSIMILE TRANSMISSION
I hereby certify that this correspondence is being transmitted by facsimile to fax number 703-872-9306 to the U.S. Patent and Trademark Office on April 21, 2005.

Signed: _____
Agnes Spence

APPEAL BRIEF TRANSMITTAL
(37 CFR 192)

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This brief is in furtherance of the Notice of Appeal filed in this case on February 23, 2005.

This application is on behalf of

☐ Small Entity ☒ Large Entity

Pursuant to 37 CFR 1.17(f), the fee for filing the Appeal Brief is:
☐ $250.00 (Small Entity) ☒ $500.00 (Large Entity)

☐ Applicant(s) hereby petition for a _____ extension(s) of time to under 37 CFR 1.136.

If an additional extension of time is required, please consider this a petition therefor.

☐ An extension for months has already been secured and the fee paid therefor of $ is deducted from the total fee due for the total months of extension now requested.

☒ Applicant(s) believe that no (additional) Extension of Time is required; however, if it is determined that such an extension is required, Applicant(s) hereby petition that such an

12/19/2005 LWONDIM1 00000022 500388 09467387
01 FC:1401 500.00 DA

RECEIVED
CENTRAL FAX CENTER
DEC 16 2005

extension be granted and authorize the Commissioner to charge the required fees for an Extension of Time under 37 CFR 1.136 to Deposit Account No. 500388.

Total Fee Due:

Appeal Brief fee	$500.00
Extension Fee (if any)	$
Total Fee Due	$500.00

☐ Enclosed is Check No. in the amount of $.

☒ Charge any additional fees or credit any overpayment to Deposit Account No. 500388, (Order No. SUN1P253). Two copies of this transmittal are enclosed.

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP

Alan S. Hodes
Reg. No. 38,185

P.O. Box 70250
Oakland, CA 94612-0250
(650) 961-8300

RECEIVED
CENTRAL FAX CENTER

DEC 16 2005

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF APPEALS

EX PARTE EDUARDO PELEGRI-LLOPART ET AL.

Application for Patent

Filed December 21, 1999

Application No. 09/467,387

FOR:

MULTI-LINGUAL TAG EXTENSION MECHANISM

APPEAL BRIEF

CERTIFICATE OF FACSIMILE TRANSMISSION
I hereby certify that this correspondence is being transmitted by facsimile to fax number 703-872-9306 to the U.S. Patent and Trademark Office on April 21, 2005.

Signed: _____
Agnes Spence

BEYER WEAVER & THOMAS, LLP
Attorneys for Applicant

TABLE OF CONTENTS

I. Real Party in Interest (37 CFR 41.37(c)((1)(i))

The real party in interest is Sun Microsystems, Inc.

II. Related appeals and interferences (37 CFR 41.37(c)(1)(ii))

There are no related appeals, interferences or judicial proceedings, except that related application no. 09/471,072 (identified in the specification as filed as an application "entitled 'MECHANISM FOR AUTOMATIC SYNCHRONIZATION OF SCRIPTING VARIABLES' attorney docket number SUN1P254/P4195") was appealed and an appeal brief filed, but the application was allowed prior to the appeal being forwarded to the Board.

III. Status of Claims (37 CFR 41.37(c)(1)(iii))

The status of the claims in this application is as follows:

Claims 1-25: cancelled.

Claims 26-41: rejected.

Claim 42: cancelled.

The claims whose rejection is being appealed are Claims 26-41.

IV. Status of Amendments (37 CFR 41.37(c)(1)(iv))

The Examiner has indicated, in the Advisory Action mailed November 10, 2004, that the amendments made after final rejection would be entered for the purposes of appeal.

That said, while claim 42 was cancelled in the response filed after final rejection, the Examiner has indicated in the Advisory Action that claims 26-42 would stand rejected for the purposes of appeal. Notwithstanding this statement, it is assumed that the cancellation of claim 42 would be entered for the purposes of appeal.

V. Summary of Claimed Subject Matter (37 CFR 41.37(c)(1)(v))

Applicant provides herewith a concise explanation of the subject matter defined in each of the independent claims involved in this appeal. This explanation is intended as a guide in distinguishing the subject matter recited in the claims from the cited reference, and it is not intended to itself limit or otherwise affect the scope of the claims in any way.

Claim 26

Claim 26 is directed to a method of processing a custom action tag in a Web Page to provide a tag library extension that can be used to implement a tag library supporting multiple scripting languages. As discussed at page 12, lines 9-10 (all citations are to the specification as filed, and not to the substitute specification), "a tag extension mechanism is a specialized sub-language that enables the addition of new or custom actions . . ."

Furthermore, at page 12, lines 14-16, it is discussed that "The tag extension mechanism of the present invention can be used from JSP pages written using any valid scripting language, although the mechanism itself only assumes a Java Run Time environment." Figure 6 illustrates an example of a tag library, including a doStart() method, a doBody() method and a doEnd() method for a custom action.

At page 14, lines 5-6, it is discussed that: "As shown in Figure 6, the code of Figure 4 can be translated using the present invention into Java code, without using a closure." Figure 4, on the other hand, illustrates how a tag can be translated (undesirably) into a closure. See page 5, line 7. That is, as discussed at page 5, lines 19, "A closure is difficult to create, however, since it is difficult to provide the requisite context. In fact, the Java programming language does not support the use of closures."

Significantly, claim 26 recites a method for processing a custom action tag where, rather than creating a complex "closure" abstraction, the evaluation of the body is "in-lined." See page 13, lines 20-21.

With that as background, we now turn to Figure 5. Figure 5 illustrates an example of the method recited in claim 26, to process a custom action tag. For example, still referring to Figure 5, the tag-handler object represents a run-time representation of a custom action tag.

In Figure 5, the do-start method is f.doStart(); the do-body method is f.doBody(); and the do-end method is f.doEnd().

Claim 26 recites, in part:

> wherein said do-start method determines:
> whether the custom action tag has a body, and
> whether there is a need to process said body when said do-start method determines that said custom action tag has a body;

Referring still to Figure 5, the f.doStart() method processes the start tag and determines if there is a body and whether there is a need to process the body (result "b").

Claim 26 further recites:

> invoking said do-body method of said tag-handler object when said invoking of said do-start method determines that there is a need to process said body of said custom action tag;
> processing, by said do-body method of said tag-handler object, said body of said custom action tag, to translate said body from a first scripting language to platform independent code that can be executed to perform actions intended by said custom action tag;
> determining by said do-body method of said tag-handler object whether further processing is required to translate the body from a first scripting language to platform independent code that can be executed to perform the actions intended by said custom action tag when said processing has been performed by said do-body method of said tag-handler object,
> repeating said processing, by said do-body method of said tag-handler object, when said do-body method of said tag-handler object determines that further processing is required; and

invoking said do-end method of said tag-handler object when said do-body method determines that no further processing is required, wherein said do-end method processes said end-tag of said custom action tag.

Figure 6 illustrates both invoking the do-body method, and the processing of the do-body method, at f.doBody(). The result "b" of b=f.doBody() in Figure 6 indicates a determination of whether further processing is required, and "if (b) goto repeat" in Figure 6 corresponds to the step of repeating do-body method processing, recited in claim 26. Finally, f.doEnd() in Figure 6 corresponds to the recitation of "invoking said do-end method."

Claim 35

Claim 35 is directed to a computer readable media. The computer readable media includes computer program code for processing a custom action tag in a Web Page to provide a tag library extension that can be used to implement a tag library supporting multiple scripting languages. Given the similarities of claim 35 to claim 26, a separate concise explanation specifically directed to claim 35 is not provided. Rather, Applicant incorporates the concise explanation provided above regarding claim 26.

Claim 40

Claim 40 is directed to a computer system. The computer system includes computer program code capable of processing a custom action tag in a Web Page to provide a tag library extension that can be used to implement a tag library supporting multiple scripting languages. Given the similarities of claim 40 to claim 26, a separate concise explanation specifically directed to claim 40 is not provided. Rather, Applicant incorporates the concise explanation provided above regarding claim 26.

VI. Grounds of Rejection (37 CFR 41.37(c)(1)(vi))

There is only one ground of rejection. Specifically, claims 26–42 are rejected under 35 U.S.C. 102(a) as being anticipated by Ale Fedorov, et al., "Professional Active Server Pages 2.0," 1998, Wrox Press Ltd (hereinafter [Fedorov98].

As discussed above in "Status of Amendments," it is assumed for the purpose of this appeal that claim 42 has been cancelled.

Argument (37 CFR 41.37(c)(1)(vii))

Summary of Argument

For a reference to anticipate the subject matter of a claim, the reference must disclose each and every feature recited in that claim. Applicant contends that [Fedorov98] does not disclose each and every feature recited in the rejected claims. (Ground 1)

Furthermore, Applicant contends that the Examiner has not even met the burden (37 CFR 1.104(c)(2)) of pointing out the particular parts of [Fedorov98] relied upon in rejecting the

claims. The Examiner's reliance on broad potions of [Fedorov98], particularly where the same broad portion is applied against multiple claim elements, does not give sufficient notice of how the Examiner is applying the reference in finding the recited subject matter to be anticipated. (Ground 2)

A. *Ground 1: Failure to Disclose Each and Every Feature*

Independent Claim 26

Applicant respectfully contends that, as the rejection is best understood by Applicant (i.e., subject to the "Request for Clarification" made by Applicant, and rejected by the Examiner as inapplicable), [Fedorov98] fails to disclose each and every feature recited in Applicant's claim 26.

Thus, [Fedorov98] does not anticipate Applicant's claim 26. Applicant's contentions are discussed in detail below.

A. Do-Start Method

The Examiner contends that an Application_onStart event handler or a Session_onStart event handler corresponds to the do-start method recited in claim 26 as being included in a tag-hander object. The Examiner cites to [Fedorov98] as follows:

> see, for example, "Application Event Handlers" on pp. 132-133, "Session Event Handlers" and "Session Variables" on pp. 138-139, and "Application and Session Events" on pp. 328-329)"

With regard to the feature of "invoking said do-start method . . . ," the Examiner's citations are the same, except that, for the recitation of what the do-start method determines

> whether the custom action tag has a body

and

> whether there is a need to process said body when said do-start method determines that said custom action tag has a body (emphasis added)

the Examiner cites as follows: "see, for example, Applications, Sessions and State' on pp. 325-341."

With regard to pp. 132-133 and pp. 138-139, these pages clearly do not disclose that the onStart handlers (Application_onStart and Session_onStart) make the determinations recited in claim 26. Rather, these pages disclose only "Insert script to be executed when the application starts" (for Application_on_Start) and "Insert script to be executed when a session starts" (for Session_onStart).

In addition, pages 328-329 disclose that the onStart events are used to initialize state. There is no disclosure in this cited portion of [Fedorov98] of determining whether there is a body and whether there is a need to process the body. Specifically, this section discloses the following with respect to the Application_onStart and Session_onStart events:

> Both of these events are used to initialize state, by setting up variables that are global for the application or for a specific user. When the first user accesses a file in our

> application, the **Application_onStart** event is triggered. This is used to initialize any application-wide global variables. When the user begins a session for the first time, the **Session_onStart** event is triggered. This is used to initialize user-specific information.
>
> The power of the **Session** object comes from the fact that it can store variables that are global to just that specific user, and so each user can have their own individual value for that variable. **Session** objects aren't always created automatically for every user when they enter our application. However, storing or accessing a variable in the Session object will create it, and fire the **Session_onStart** event. We can force to sessions to always be created as soon as a visitor enters our application by writing code in **global.asa** to respond to this event.

We note that firing the Session_onStart event when a variable in the Session object is created, based on storing or accessing the variable, is not the same as determining whether there is a body and whether there is a need to process the body.

This leaves pp. 325-341, generically, of [Fedorov98] as the sole remaining possible cited source for disclosing the determinations recited in claim 26 as being carried out by the do_start method (i.e., whether there is a body and whether there is a need to process the body). That is, we have specifically distinguished the disclosure at pp. 132-133, pp. 138-139, and pp. 328-329; and the Examiner cites pp. 325-341 generally as support for the contention that [Fedorov98] discloses the determining steps of the do-start method.

Applicant has made a good faith attempt to find a disclosure of the Application_onStart and Session_onStart handler making the determinations recited in claim 26. Besides a general disclosure that the onStart handlers are uses to initialize variables, the only other disclosure within pp. 325-341 that Applicant can find of processing within an onStart handler is at pages 334 and 336.

The disclosure at pages 334 and 336 disclose creating an instance of an object if one does not already exist. For example, at page 334, there is the following code section disclosed as being part of a Session_onStart routine:

```
If IsEmpty(Application("Object")) Then
    Set Application ("Object") = Server.CreateObject ("global.connection")
    Application ("ObjectCount") = 0
End If
```

At page 336, there is a similar code section:

```
If IsEmpty(Application("myData")) Then
    Application ("myData") =strTheValue
End If
```

Neither of these disclosures of onStart hander processing disclose determining whether there is a body and whether there is a need to process the body.

B. Do-Body Method

The Examiner also contends that [Fedorov98] discloses do-body method processing "to translate said body from a first scripting language to platform independent code that can be

executed to perform actions intended by said custom tag." The Examiner specifically refers to " 'The Response Object' on pp. 113-125 and 'The Server Object' on pp. 125-131"). Applicant can find no disclosure of this recited do-body method processing in the cited portions of [Fedorov98].

In particular, there is no disclosure in the cited portions of [Fedorov98] (as best understood by Applicant, without the benefit of a more specific contention by the Examiner, requested and refused) of a "body" of a custom action tag (i.e., wherein "said custom action tag further capable of having a body between said start-tag and said end-tag") being translated from a first scripting language to platform independent code.

In fact, the Examiner has not even pointed out where the cited portions of [Fedorov98] are contended to disclose a "body" (again, between a start-tag and an end-tag) at all, and Applicant respectfully contends that the cited portions of [Fedorov98] do not include such a disclosure.

Furthermore, there is no disclosure in the cited portions of [Fedorov98] (again, as best understood by Applicant) of "determining by said do-body method of said tag-handler object whether further processing is required to translate the body from a first scripting language to platform independent code . . ."

Independent Claim 35

Applicant has specifically pointed out several features recited in claim 26 that are not present in the cited portions of [Fedorov98]. With respect to claim 35, similar features are recited in that claim. Thus, in distinguishing the recitations of claim 35 from the cited portions of [Fedorov98], Applicant incorporates the contentions made above to distinguish the features of claim 26 from the cited portions of [Fedorov98].

Independent Claim 40

Applicant has specifically pointed out several features recited in claim 26 that are not present in the cited portions of [Fedorov98]. With respect to claim 40, similar features are recited in that claim. Thus, in distinguishing the recitations of claim 40 from the cited portions of [Fedorov98], Applicant incorporates the contentions made above to distinguish the features of claim 26 from the cited portions of [Fedorov98].

Dependent Claim 32

Claim 32 depends from claim 31, and further recites that the do-end method "facilitates cleanup and update of said context page; and determines whether further processing is required for another custom action tag."

The Examiner contends that the disclosure of "Scripting Objects" (at pp. 147-170 of [Fedorov98]) discloses "facilitates cleanup and update of said context page" and the disclosure

of "The Response Object" (at pp. 113-125 of [Fedorov98] discloses "determines whether further processing is required for another custom action tag."

With regard to "The Response Object," the Examiner cited these same pages as generally disclosing the "do-end method." At pp. 113-125, the "Response Object" is disclosed, for example, as "Whereas the Request object is totally concerned with what is coming to our server from the browser, the Response object, not surprisingly, handles all the stuff we want to send back to it." [Fedorov98], p. 113. At p. 114, it is described that the "Response Object implements a single collection, cookies, five properties, and eight methods."

None of the cookies, properties or methods are described as "determines whether further processing is required for another custom action tag."

Specifically at page 115, the "interface elements" of the Response Object are characterized into six groups, none of which are concerned with "determines whether further processing is required for another custom action tag." The six groups listed on page 115 are:

1. Inserting information into a page.
2. Sending cookies to the browser.
3. Redirecting the browser.
4. Buffering the page as it is created.
5. Setting the properties of a page.
6. Checking the client connection.

Thus, as best understood by Applicant, the disclosure of the "Response Object" does not meet the recitation of a do-end method "determines whether further processing is required for another custom action tag."

Dependent Claim 33

Dependent claim 33 recites that the "do-body" method comprises "a do-init-body method that operates to initialize and pre-process a body-evaluation object that provides an abstraction from a scripting language used to implement said custom action tag." Dependent claim 33 further recites that the "do-body" method comprises "a do-after-body method that operates to evaluate said body-evaluation object at least once to translate the body from a first scripting language to platform independent code that can be executed to perform the actions intended by said custom action tag."

For each of these elements, the Examiner again generically cites to "The Response Object" on pp. 113-125. As discussed above with respect to dependent claim 32, the "Interface Elements" of the Response Object are characterized into six groups. None of these groups appear to Applicant to be concerned with "operates to initialize and pre-process a body evaluation object . . ." or "operates to evaluate said body-evaluation object at least once to translate the body from a first scripting language to platform-independent code . . ."

Thus, as best understood by Applicant, the disclosure of the "Response Object" does not meet the recitation set forth in claim 33.

Summary (Failure to Disclose each and every Feature)

For the reasons discussed above in detail, the [Fedorov98] reference fails to disclose each and every element of the claims specifically discussed above. For at least this reason, the Examiner's anticipation rejection is improper and should be overturned.

B. *Ground 2: Examiner has Not Clearly Explained Pertinence of Reference*

The Examiner has not followed the requirement of 37 CFR 1.104(c)(2) which requires, in part, "When a reference is complex or shows or describes inventions other than that claimed by the applicant, the particular part relied on must be designated as nearly as practicable. The pertinence of each reference, if not apparent, must be clearly explained and each rejected claim specified." (emphasis added) In the final office action (as well as in the initial office action), the pertinence of the cited [Fedorov98] reference is clearly not apparent and, thus, the Examiner has not made a proper rejection. What a detailed designation, Applicant is not on notice or to what is the rejection.

For some of the features, the Examiner cites to portions of [Fedorov98] that are up to seventeen pages, without particularly citing where these portions allegedly disclose a particular feature. Furthermore, the same large portions of the [Fedorov98] reference are applied to multiple distinct claim features, without particularly pointing out which specific part of the portion applies to one distinct feature and which other specific part of the portion applies to another distinct feature.

Independent Claim 26

For example, the Examiner contends that pages 325-341 (seventeen pages) of [Fedorov98] discloses:

> wherein said do-start method determines
> whether the custom action tag has a body, and
> whether there is a need to process said body when said do-start method determines that said custom action tag has a body.

The Examiner similarly points to the same (seventeen page) portion of [Fedorov98] for the alleged disclosure of:

> invoking said do-start method of said tag-handler object when invoking of said do-start method determines there is a need to process said body of said custom action tag.

The Examiner also contends that both a thirteen page portion (pages 113-125) and a seven page portion (pages 125-131) of [Fedorov98] discloses the three features of:

> processing, by said do-body method of said tag-handler object, said body of said custom action tag, to translate said body from a first scripting language to

platform independent code that can be executed to perform actions intended by said custom action tag,

determining by said do-body method of said tag-handler object whether further processing is required to translate the body from a first scripting language to platform independent code that can be executed to perform the actions intended by said custom action tag when said processing has been performed by said do-body method of said tag-handler object,

and

repeating said processing, by said do-body method of said tag-handler object, when said do-body method of said tag-handler object determines that further processing is required.

It can be thus seen that the Examiner's allegations with respect to claim 26 are insufficient to meet the Examiner, burden.

Independent Claim 35

The allegations against independent claim 26 are incorporated by the Examiner in rejecting independent claim 35. Thus, Applicant incorporates the arguments set forth above with respect to claim 26 in refutation of the rejection of claim 35.

Independent Claim 40

The allegations against independent claim 26 are incorporated by the Examiner in rejecting independent claim 40. Thus, Applicant incorporates the arguments set forth above with respect to claim 26 in refutation of the rejection of claim 40.

Dependent Claims 27, 28, 29 and 31

The Examiner's allegations relative to others of the claims are similarly deficient to meet the Examiner's burden. As just some examples, the features recited in claim 27 are alleged to also be disclosed at pages 113-125 of [Fedorov98], as are the features recited in claims 28, 29 (also citing the twenty four page portion of [Fedorov98] from pages 147-170) and 30 (also citing pages 147-170). The features recited in claim 31 are alleged to be disclosed at the twenty four page portion at pages 147-170 also.

Without specifically listing the number of pages recited against the features of each claim, it can be seen from the Office Action that multiple pages of the [Fedorov98] disclosure are cited against each of the claims, with many of these multiple page citations being simultaneously cited against multiple ones of the features.

C. **Summary (Examiner has Not Clearly Explained Pertinence of Reference)**

Admittedly, the CFR section that requires the Examiner to cite a particular part of a reference "as nearly as practicable," and to clearly explain how that particular part of the reference applies to a specific claim, does not specify how to determine that a "reference is complex or shows or describes inventions other than that claimed by Applicant."

Where as here, though the Examiner cites sections of a 991 page reference (page count according to Amazon.com), the reference can clearly be presumed to be "complex" and describing a lot of subject matter. Furthermore, where the cited sections themselves are up to twenty four pages and are each cited for multiple features, the Examiner clearly has not cited to a particular part of the reference "as nearly as practicable." As a result, Applicant cannot be considered to be properly on notice as to what is the rejection.

The Examiner has clearly failed to comply with 37 CFR 1.104(c)(2), and the rejection based on [Fedorov98] is improper and should be withdrawn for at least this reason.

VII. CONCLUSION

It is respectfully submitted, as [Fedorov98] is understood by Applicant without the benefit of specific designations by the Examiner, that [Fedorov98] does not disclose each and every feature recited in the rejected claims. Applicant has specifically pointed out, using the language of the claims, how the cited portions of [Fedorov98] do not disclose particular features recited in the claims.

It is once again additionally respectfully submitted that the Examiner has failed to comply with 37 CFR 1.104(c)(2), and Applicant respectfully renews the request for a more specific designation of the portions of [Fedorov98] relied upon in rejecting the claims, in the event the Examiner does not allow the application.

Applicant respectfully requests that the anticipation rejection over [Fedorov98] be overturned and that the application be passed to allowance.

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP

Alan S. Hodes
Reg. No. 38,185

P.O. Box 70250
Oakland, CA 94612-0250
(650) 961-8300

VIII. APPENDIX OF CLAIMS

Claims 1-25 (Cancelled)

26. (Previously Presented) A method of processing a custom action tag in a Web Page to provide a tag library extension that can be used to implement a tag library supporting multiple scripting languages, said custom action tag providing textual instructions for performing actions in said Web Page, said custom action tag being capable of customization to perform customized actions and including a start-tag and an end-tag, said custom action tag being further capable of having a body between said start-tag and said end-tag, said method comprising;

providing, for said custom action tag, a tag-handler object that represents a run-time representation of said custom action tag, said tag-handler object including:

a do-start method for processing said start-tag of said custom action tag,

a do-body method for processing said body of said custom action tag, and

a do-end method for processing of said end-tag of said custom action tag;

invoking said do-start method of said tag-handler object to process said start-tag of said custom action tag, wherein said do-start method determines:

whether the custom action tag has a body, and

whether there is a need to process said body when said do-start method determines that said custom action tag has a body;

invoking said do-body method of said tag-handler object when said invoking of said do-start method determines that there is a need to process said body of said custom action tag;

processing, by said do-body method of said tag-handler object, said body of said custom action tag, to translate said body from a first scripting language to platform independent code that can be executed to perform actions intended by said custom action tag;

determining by said do-body method of said tag-handler object whether further processing is required to translate the body from a first scripting language to platform independent code that can be executed to perform the actions intended by said custom action tag when said processing has been performed by said do-body method of said tag-handler object,

repeating said processing, by said do-body method of said tag-handler object, when said do-body method of said tag-handler object determines that further processing is required; and

invoking said do-end method of said tag-handler object when said do-body method determines that no further processing is required, wherein said do-end method processes said end-tag of said custom action tag.

27. (Previously Presented) A method as recited in claim 26, wherein said method further comprises:

creating at runtime when said platform independent code is to be executed, by said tag-handler, one or more objects that said Web Page requires;

storing at runtime when said platform independent code is to be executed, by said tag-handler at runtime, said one or more created objects into a pageContext object, thereby allowing the one or more objects to be retrieved at runtime.

28. (Previously Presented) A method as recited in claim 26,

wherein said invoking of said do-start method of said tag-handler object provides said do-start method with one or more attribute values associated with said custom action tag, and

wherein said method further comprises:

providing a page context object that provides runtime data including run-time values for one or more attributes associated with said custom action tag; and

interacting, by said do-start method, with said page context object to perform appropriate functionality with respect to said attribute values.

29. (Previously Presented) A method as recited in claim 26, wherein said processing of said custom action tag by said do-body method comprises:

evaluating said body of said custom action tag as a stream of bytes that can be represented as a body-evaluation object, said body-evaluation object providing an abstraction from various scripting languages that can be used to implement said custom action tag.

30. (Previously Presented) A method as recited in claim 29, wherein a body-evaluation object is implemented using a buffer.

31. (Previously Presented) A method as recited in claim 29, wherein said evaluating of said custom action tag by said do-body method further comprises:

providing said do-body method with said body-evaluation object;

determining by said do-body method whether to insert said stream of bytes into another stream of bytes;

inserting, by said do-body method, said stream of bytes into another stream of bytes, when said do-body method determines that said stream of bytes should be inserted into another stream of bytes; and

converting, by said do-body method, said stream of bytes to a string when said do-body method determines that said stream of bytes should not be inserted into another stream of bytes.

32. (Previously Presented) A method as recited in claim 31, wherein said do-end method:

facilitates cleanup and update of said context page; and

determines whether further processing is required for another custom action tag.

33. (Previously Presented) A method as recited in claim 26, wherein the do-body method of said tag-handler object comprises:

a do-init-body method that operates to initialize and pre-process a body-evaluation object that provides an abstraction from a scripting language used to implement said custom action tag; and

a do-after-body method the operates to evaluate said body-evaluation object at least once to translate the body from a first scripting language to platform independent code that can be executed to perform the actions intended by said custom action tag.

34. (Previously Presented) A method as recited in claim 26, wherein said Web Page is implemented on a server that supports a platform independent programming language.

35. (Previously Presented) A computer readable media including computer program code for processing a custom action tag in a Web Page to provide a tag library extension that can be used to implement a tag library supporting multiple scripting languages, said custom action tag providing textual instructions for performing actions in said Web Page, said custom action tag being capable of customization to perform customized actions and including a start-tag and an end-tag, said custom action tag being further capable of having a body between said start-tag and said end-tag, said readable media comprising;

computer program code for providing, for said custom action tag, a tag-handler object that represents a run-time representation of said custom action tag, said tag-handler object including:

a do-start method for processing said start-tag of said custom action tag,

a do-body method for processing said body of said custom action tag, and

a do-end method for processing of said end-tag of said custom action tag;

computer program code for invoking said do-start method of said tag-handler object to process said start-tag of said custom action tag, wherein said do-start method determines:

whether the custom action tag has a body, and

whether there is a need to process said body when said do-start method determines that said custom action tag has a body;

computer program code for invoking said do-body method of said tag-handler object when said invoking of said do-start method determines that there is a need to process said body of said custom action tag;

computer program code for processing, by said do-body method of said tag-handler object, said body of said custom action tag, to translate said body from a first scripting language to platform independent code that can be executed to perform actions intended by said custom action tag;

computer program code for determining by said do-body method of said tag-handler object whether further processing is required to translate the body from a first scripting language to platform independent code that can be executed to perform the actions intended by said custom action tag when said processing has been performed by said do-body method of said tag-handler object,

computer program code for repeating said processing, by said do-body method of said tag-handler object, when said do-body method of said tag-handler object determines that further processing is required; and

computer program code for invoking said do-end method of said tag-handler object when said do-body method determines that no further processing is required, wherein said do-end method processes said end-tag of said custom action tag.

36. (Previously Presented) A computer readable medium as recited in claim 35,

wherein said invoking of said do-start method of said tag-handler object provides said do-start method with one or more attribute values associated with said custom action tag, and

wherein said computer readable medium further comprises:

computer program code for providing a page context object that provides runtime data including run-time values for one or more attributes associated with said custom action tag; and

computer program code for interacting, by said do-start method, with said page context object to perform appropriate functionality with respect to said attribute values.

37. (Previously Presented) A computer readable medium as recited in claim 35, wherein said processing of said custom action tag by said do-body method comprises:

evaluating said body of said custom action tag as a stream of bytes that can be represented as a body-evaluation object, said body-evaluation object providing an abstraction from various scripting languages that can be used to implement said custom action tag.

38. (Previously Presented) A computer readable medium as recited in claim 37, wherein the body-evaluation object is implemented using a buffer.

39. (Previously Presented) A computer readable medium as recited in claim 38, wherein said evaluating of said custom action tag by said do-body method further comprises:

providing said do-body method with said body-evaluation object;

determining by said do-body method whether to insert said stream of bytes into another stream of bytes;

inserting, by said do-body method, said stream of bytes into another stream of bytes, when said do-body method determines that said stream of bytes should be inserted into another stream of bytes; and

converting, by said do-body method, said stream of bytes to a string when said do-body method determines that said stream of bytes should not be inserted into another stream of bytes.

40. (Previously Presented) A computer system for processing a custom action tag in a Web Page to provide a tag library extension that can be used to implement a tag library supporting multiple scripting languages, said custom action tag providing textual instructions for performing actions in said Web Page, said custom action tag being capable of customization to perform customized actions and including a start-tag and an end-tag, said custom action tag being further capable of having a body between said start-tag and said end-tag, said computer system comprising;

at least one central processing unit;

memory; and

a computer program operating on said at least one central processing unit, wherein said computer program is capable of:

providing, for said custom action tag, a tag-handler object that represents a run-time representation of said custom action tag, said tag-handler object including:

a do-start method for processing said start-tag of said custom action tag,

a do-body method for processing said body of said custom action tag, and

a do-end method for processing of said end-tag of said custom action tag;

invoking said do-start method of said tag-handler object to process said start-tag of said custom action tag, wherein said do-start method determines:

whether the custom action tag has a body; and

whether there is a need to process said body when said do-start method determines that said custom action tag has a body;

invoking said do-body method of said tag-handler object when said invoking of said do-start method determines that there is a need to process said body of said custom action tag;

processing, by said do-body method of said tag-handler object, said body of said custom action tag, to translate said body from a first scripting language to platform independent code that can be executed to perform actions intended by said custom action tag;

determining by said do-body method of said tag-handler object whether further processing is required to translate the body from a first scripting language to platform independent code that can be executed to perform the actions intended by said custom action tag when said processing has been performed by said do-body method of said tag-handler object;

repeating said processing, by said do-body method of said tag-handler object, when said do-body method of said tag-handler object determines that further processing is required; and

invoking said do-end method of said tag-handler object when said do-body method determines that no further processing is required, wherein said do-end method processes said end-tag of said custom action tag.

41. (Previously Presented) A computer system as recited in claim 40, wherein said Web Page is implemented on a server that supports a platform independent programming language.

42. (Cancelled)